REMARKS/ARGUMENTS

In the Office Action, all of the pending claims 1-18 were rejected under 35 U.S.C. 102(b) as being anticipated by Lewis (U.S. Patent No. 5,812,394). Additionally, claims 6, 10 and 16 were objected to because of certain informalities, namely certain typographical errors.

In response to the objections to claims 6, 10 and 16, the Applicants have amended each of those claims to correct the typographical errors noted by the Examiner. The Applicants appreciate the Examiner's careful review of the claims. The Applicants have also amended claim 1 slightly to correct an antecedent basis problem.

With respect to the rejections of claims 1-18, the Applicants respectfully traverse these rejections for at least the reasons discussed below.

Further in response to the Office Action, the Applicants have added new claims 19-30 as shown above. The Applicants respectfully submit that these claims do not add new matter and are supported by the Specification at, for example, page 11, line 10 through page 14, line 19.

Rejections Under 102(b)

Despite the comments in the Office Action, the Applicants respectfully traverse the rejections of claims 1-18. The Applicants submit that Lewis fails to teach a library system that operates in the manner recited in independent claims 1 and 16. In particular, the Applicants submit that Lewis fails to show at least the following features recited in claim 1:

> a second program builder accepting information about the first linking process, and user input, to create a second portion of the control program from second program fragments taken from the same files of the first program fragments used in the first portion of the control program, the second program builder renaming the tags of the control variables of the second program fragments to comport with the renaming of the tags of the control variables of the first portions by the first program builder;

Similarly, the Applicants submit that Lewis fails to show at least the following features recited in claim 16:

> a second program builder accepting information identifying the files of the library manager from which the instances of the first program fragments originated to display to a user second program fragments related to each instance of the first program fragments according to common library files, and accepting user input to select among the displayed second

program fragments to create a second portion of the control program from second program fragments, the second program builder renaming the tags of the control variables of the second program fragments to comport with the renaming of the tags of the control variables of the first portions by the first program builder;

In the Office Action, the Examiner points to the following excerpt of the Lewis patent (column 12, lines 36-52 and column 28, lines 31-46) as teaching the above-mentioned limitations of pending claims 1 and 16:

> The development system includes a device diagramming component for describing a physical description of a facility and a logical definition of a control scheme for the facility. The device diagramming component includes a mode for selecting device symbols representative of equipment or control functions used in facilities. The device symbols are selected from an object-oriented repository containing a plurality of device symbols and device objects. Certain types of device symbols relate to device objects containing logical instructions and configuration information relating to the represented equipment or control functions. The device diagramming component also includes a mode for interrelating in a graphical manner the selected device symbols and their corresponding device objects into a device diagram representing the physical description of the facility and the logical definitions of the control scheme of the facility. ...FIG. 13 illustrates the type of configuration information that could be supplied for a discrete device upon insertion into a diagram. Note that the user chooses the template on which to base the new device, Pump-1, and specifies the name of the device, PMP-1236. When the user finished filling out this dialog and clicks OK, he/she has specified all the configured parameters associated with the new device. This might include pages of logic, tag definitions, graphic symbols, and the dynamic behavior of graphic symbols. ...Using object oriented techniques, UCOS handles all the overhead of making the device unique including making each tag name unique, as is illustrated in the FIG. 14. For example, Pump-1.Running is automatically changed to PMP-1236.Running.

Nevertheless, the Applicants disagree that this passage teaches the above-identified features recited in pending claims 1 and 16. Indeed, the Applicants fail to understand the relevance of this passage to the "second program builder" features of claims 1 and 16.

The Applicants' invention relates to a system that facilitates the creation of complex programs that employ program fragments that are stored in a library. The system operates by providing a graphical user interface on which a user is capable of arranging and connecting icons to represent a real-life system (for example, an industrial control system) for which a control program is to be created. Typically, the icons are

connected in a hierarchical manner representing the hierarchical ordering of components in the real-life system. Thus, an "area model" representing the real-life system is formed.

Once the area model has been formed, it is displayed alongside the contents of a library. The library stores generalized program fragments that can be, for example, program fragments for performing logical operations or for operating a graphical user interface. The user provides inputs (for example, drag and drop commands) indicating which of the generalized program fragments should be associated with which portions of the real-life system by associating certain program fragments with certain icons in the area model. In accordance with the present invention, as these inputs are provided, the "program builders" automatically instantiate these generalized program fragments in a manner that modifies the generalized input and output variables of the program fragments so that they pertain particularly to the specified elements of the area model with which they have been associated by the user. As a result of this variable modification process, the generalized program fragments become specific program fragments that pertain to the specific real-life system components represented by the specified elements of the area model. Further as a result of this variable modification process, the various program fragments with their modified variables become interconnected so as to produce a complete control program.

The Applicants' claimed system further envisions that, in many circumstances, there may be more than one generalized program fragment that are related to one another, For example, program fragments can share the same input and output variables even though one of the program fragments relates to control logic and another of the program fragments relates to a graphical user interface. In accordance with the present invention, program fragments that are related can be stored together, in the library, within corresponding files for those groups of program fragments. By storing these related program fragments in this way, the relationships between related program fragments are apparent to the system when creating a control program.

Additionally, the Applicants' claimed system envisions that, in many circumstances, some program fragments may be instantiated at a certain point in the creation of an overall control program, while other, related program fragments may be instantiated at a later point in the creation process. The Applicants' claimed system facilitates the creation of an overall control program because, as it modifies the variables of program fragments during instantiation, it maintains certain characteristics of those

variables that indicate the library files from which those program fragments came. Consequently, when related program fragments are instantiated at different times during the creation of the control program, the system is able to automatically recognize related program fragments in the library that relate to earlier-instantiated program fragments, and is able to easily and automatically instantiate the related program fragments so that they are properly integrated with the earlier-instantiated program fragments in the control program.

For example, a control program often is created by instantiating program fragments relating to logical operations first and then by instantiating related program fragments concerning the graphical user interface next. In accordance with the present invention as recited in claims 1 and 16, because related program fragments are stored in the library in the same files, the "second program builder" is able to recognize when a selected graphical user interface program fragment is related to a logical operation program fragment that has already been instantiated within the program. Using this information, the second program builder is able to automatically modify the variables of the graphical user interface program fragments so that they conform to the variables of the already-instantiated logical operation program fragments. This facilitates interconnection both between the related logical operation and graphical user interface program fragments, and also between the different graphical user interface program fragments associated with different portions of the real-life system as represented by different elements of the area model.

Thus, the claimed library system of claims 1 and 16 has a "library manager" that stores, in multiple "unique files", "first and second program fragments" that are related in that they have "shared control variables" having "common tags". Following the creation of a first portion of a control program by a first program builder, a "second program builder" creates a second portion of the control program from the second program fragments taken from the files in which the first and second program fragments are stored. Because the related first and second program fragments are stored in the same files, the second program builder is able to recognize the related files and thereby is capable of automatically "renaming the tags of the control variables of the second program fragments to comport with the renaming of the tags of the control variables of the first portions".

These features of claims 1 and 16 involving the recognition of related program fragments based upon the manner in which the fragments are stored, and using such relatedness information to rename the control variables of the related program fragments in the same manner by first and second program builders, are not addressed by the excerpts of the Lewis patent identified in the Office Action. Indeed, the excerpts of the Lewis patent do not appear to discuss first and second program builders, do not appear to discuss related program fragments having shared control variables being stored in dedicated files, and do not appear to discuss the automatic renaming of control variables of secondary program fragments in a manner consistent with the renaming of control variables of primary program fragments based upon the storage of those related program fragments in corresponding dedicated files.

Therefore, for at least these reasons, the Applicants submit that independent claims 1 and 16, as well as claims 2-15 and 17-18 depending therefrom, are allowable over the Lewis patent.

* * *

Conclusion

Given the Applicant's Remarks and Amendments, the Applicant respectfully requests reconsideration and allowance of the present Application.

The Applicant wishes to invite the Examiner to telephone the Applicant's attorney at the number listed below if discussion with the Applicant's attorney would be of assistance to the Examiner or further the prosecution of the present Application.

Respectfully submitted,
Randall A. Havner et al.

By: _____

John T. Pienkos
Reg. No. 42,997
Quarles & Brady
411 E. Wisconsin Ave., Suite 2550
Milwaukee WI 53202-4497
(414) 277-5777

QBMke#5443039.1